LR(0)

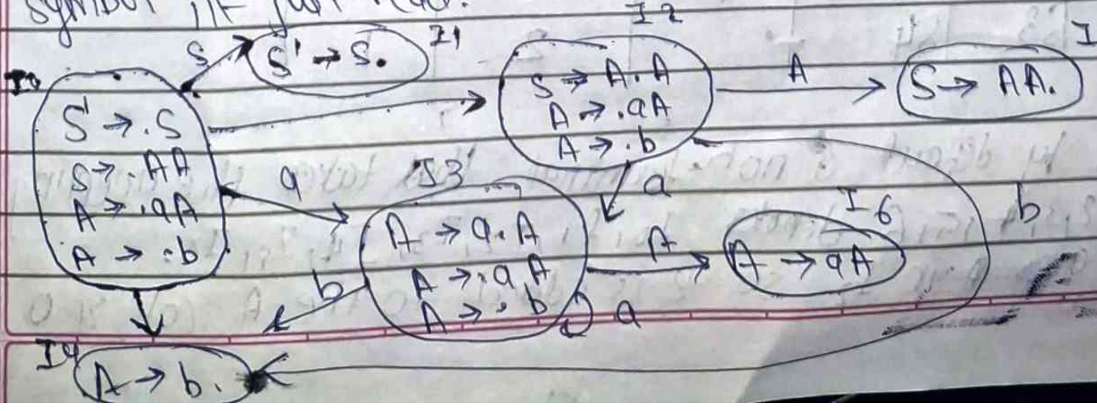
- It is a type of eficient bottom-up parser that can be used for a large class of context free grammar. The technique used is called LR(0) parsing
- L stands for left to right scanning, R stands for right most derivation in reverse. , 0 stands for no. of input symbols symbols of lookahead.

- A CFG is set of rules that describe how to generate string for a language.
- Given Grammar ⇒ S→ A A , A → a A | b

AG * If G is a grammar with starting symbol S, then G' (augmented grammar for G) is a grammar with a new starting symbol S' & productions S'→ . S
- The purpose of this new starting production is to indicate the parser when it should stop parsing.
- three rules of augmented gramer, with help of eg S→ AA
  1) S→ .AA , the '.' before A indicates the string hasnt been parsed. yet
  2) S→ A.A , the '.' after the first a indicate that it has been parsed and the next a will be parsed.
  3) S → aa. , it shows that the string has already been parsed.

* AG of given grammar: $S' \rightarrow .S$ ⓪, $S \rightarrow .AA$ ①, $A \rightarrow .aA$ ②, $A \rightarrow .b$ ③

Rule of Goto * It refers to function in bottom-up parser that determine the next state of parser based on the current state & the input symbol it just read.



Rules :- 1) If any NT has '.' preceding it, we've to wri all it's production & add '.' preceeding each of it's productions.

2) from each state to the next state, '.' shifts one place to

Steps :- 1) In given figure, I_0 contains Augmented Grammar

2) I_0 goes to I_1, when '.' of ⓪ is shifted towards right ($S' \to S.$) . the state is accepted . S is seen by compiler.

3) I_0 goes to I_2 when '.' of ① shifted right ($S \to A.A$) A is seen.

4) I_0 goes to I_3 when '.' of ② is shifted right. ($A \to a.A$) a is seen

5) I_0 goes to I_4 when '.' of ③ shifted right ($A \to b.$) b is seen.

6) I_2 goes to I_5 when '.' of ① shifted right ($S \to AA.$) A is seen

7) I_2 goes to I_3 when '.' ② shifted right ($A \to a.A$) a is seen

8) I_2 goes to I_4 when '.' of ③ shifted right ($A \to b.$) b is seen

9) I_3 goes I_4 when '.' of ③ shifted right ($A \to b.$) b is seen

10) I_3 -"- I_6 -"- ② -"- ($A \to aA.$) A is seen

11) I_3 -"- I_3 -"- ② -"- ($A \to a.A$) . a is seen.

* Parsing table :-

	Action			Goto	
S	a	b	$	A	S
1	–	–	Accepted	–	–
2	S3	S4	–	5	–
3	S3	S4	–	6	–
4	R3	R3	R3		–
5	R1	R1	R1	–	–
6	R2	R2	R2	–	
0	S3	S4	–	2	1

Steps 1) $ is by default a non-terminal that takes the acepting state.

2) 0, 1, 2, 3, 4, 5, 6 denotes $I_0, I_1, I_2, I_3, I_4, I_5, I_6$.

3) I_0 gives A in I_2 so 2 is added to the A col & 0 rows,

4) I_0 givs S in I_1, so 1 is added to the S column and 1 row.

5) Similarly s is written in A colm and 2nd row, 6 is written in A column and 3 rows.

6) I_0 gives a in I_3 so S3 (shift 3) is added to the 'a' column & 0 row,

7) I_0 give b in I_4, so S4 (shift 4) is added to the 'b' col & 0 row.

8) similarly, S3 (shift 3) is added on 'a' colm & 2, 3 rows.
→ S4 is added on 'b' column & 2, 3 rows.

9) I_4 is reduced state as '.' is at the end. I_4 is the 3rd production of grammar. so write r3 in terminals.

10) I_5 is reduced state as '.' is at the end. I_5 is the 1st production of grammar. so write r1 in terminals.

11) I_6 is reduced state as '.' is at end. I_6 is 2nd production of grammar. so write r2 in terminals.

* Parsing string abab.

\$	abab\$	Shift I_0
\$0	abab\$	Shift 3 (0A sathi → S3)
\$0a3	bab\$	Shift 4 (3b sathi → S4)
\$0a3b4	ab\$ 4a is reduced to 3	Reduce 3 A→b (reduce 2 char) & shift 6 → A6
\$0a3A6	ab\$	Reduce 2 A→aA (4 -u-) & shift. A2
\$0A2	ab\$	Shift 3
\$0A2a3	b\$	Shift 4
\$0A2a3b4	\$	Reduce 3 A→b (remove 2) & shift A6
\$0A2a3A6	\$	Reduce 2 A→aA (4) & shift. A5
\$0A2A5	\$	Reduce 1 S→AA (4) & shift S1.
\$0S1	\$	Accepted